ActiveX Control Simplifies Instrument Programming

Werner Haussmann and Alicia Viskoe
Hewlett-Packard, Loveland, CO

Use this downloadable ActiveX custom control to operate IEEE 488 instruments. And learn how to use a VB chart to plot data.

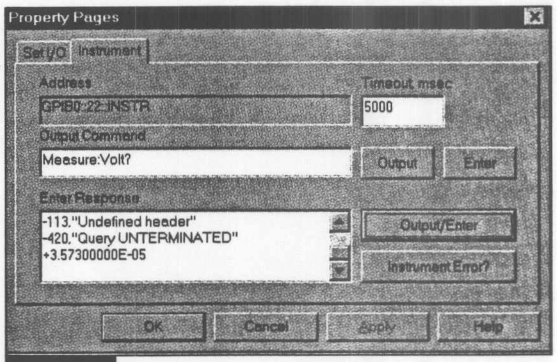
When you take measurements on your lab bench, you often need to record and display your data. Writing a program to capture and plot data can be tedious. To make Visual Basic (VB) programming easier, we've created an ActiveX control that simplifies programming for instruments over an IEEE 488 interface.

The control lets you send ASCII instrument commands to an IEEE 488 card's driver. Instead of a line of code that reads CALL ibwrt(22, "waveform:data?"), you can write a line of code like scope.output "waveform:data?" We've made the control available for free downloading through *Test & Measurement World*'s Web site: *www.tmworld.com*. Click on "Software."

Whenever you develop an application that controls instruments, you should start by checking for communications between your computer and your instruments. If you don't do this and you later develop a communication problem, you won't know if the problem is in your code, the instrument setup, or the IEEE 488 interface card, or even if you forgot to connect the IEEE 488 cable. So, we added a feature to the control that lets us check communications with instruments while VB is still at the design mode. With this control, we can check communications without running the application we're developing.

We'll show you how to use the control to establish communications with your instruments. We'll also show you how to use Microsoft's VB chart control to plot data. Our example uses an HP 54602B oscilloscope, but you can apply our programming concepts to any scope—or any other instrument, for that matter.

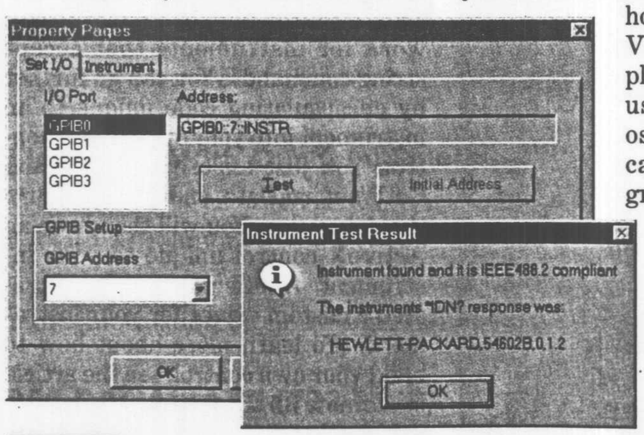
We'll assume you know how to configure the scope's sensitivity or trigger settings. If you need a refresher on how to set up a scope under computer control, see the article referenced in footnote 1.



Address: GPIB0::22::INSTR
Timeout, msec: 5000
Output Command: Measure:Volt?
Enter Response:

```
-113,"Undefined header"
-420,"Query UNTERMINATED"
+3.57300000E-05
```

FIGURE 2 This Property Page lets you communicate with the instrument during design time. The Output/Enter button executes the Output command and if a query is sent, executes the Enter command.



I/O Port: GPIB0 (GPIB1, GPIB2, GPIB3)
Address: GPIB0::7::INSTR
GPIB Address: 7

Instrument found and it is IEEE488.2 compliant
The instruments *IDN? response was:
HEWLETT-PACKARD,54602B,0,1.2

FIGURE 1 An ActiveX control lets you set the I/O address from a Property Page and lets you test the connection during development.

The following examples use VB 5.0 professional or better and a HP 54600 series scope with an IEEE 488 module. You'll also need an IEEE 488 interface card from either Hewlett-Packard or National Instruments.

Getting the Control

To use the control, your computer needs to have Windows 95 and OSR2 or Internet Explorer 4.0 or later installed.[2] Your first step is to download the control, called TMWControl, from the *T&MW* Web site. Unzip the file and run the setup.exe program. Open VB, then select "New" and "Standard EXE" to start a new project. A blank form will appear. Be sure that the Toolbox is visible. If not, select the toolbox from the "View" menu. Add the "TMW Instrument I/O Control" to your toolbox from the Components dialog box (right click on the toolbox, or use Ctrl-T).

Put the control on a form. VB will name the control TMWControl1, but we changed the control's name to "scope" for this article. Each instance of the control can communicate with just one instrument, so you must add one instance of the control to your VB project for each instrument you have. Give each control a different—but descriptive—name like scope, DMM, or FunctionGen.

Now, you can test communications between your PC and your instrument right from the VB form. To communicate with your instrument, right-click on the control and select Properties. You'll get a set I/O and instrument property page. Once you set up the address for the GPIB card and the instrument, click on the Test button. You should see the Instrument Test Result screen (**Fig. 1**) that contains the instrument's ID—if your instrument is IEEE 488.2 compliant. If your instrument isn't IEEE 488.2 compliant, you must use the Instrument page to verify communications.

With communications established, you can try out some instrument commands in the "Instrument" Property Page. **Figure 2** shows where you type an instrument command for output to the instrument and where the Enter button returns a response from the instrument.

Listing 1. This code returns and scales the waveform data from the instrument using the provided control. It demonstrates sending strings, retrieving parsed numeric data, and retrieving IEEE 488 block data. (The complete control is available at *www.tmworld.com.* Click on "Software".)

```
Sub GetWaveformData(Channel As Long,
Points As Long, time, data)
    ' Gets the waveform data from the HP546xx
    ' scope given the channel and number of points

    Dim Preamble(10) As Double
    Dim ydata As Variant
    Dim address As String
    Dim strChannel As String
    Dim i As Long

    scope.address = txtAddress.Text
    ' set scope for number of points and then
    ' get the preamble and return waveform
    ' as byte data
    strChannel = "Channel" & Format$(Channel)
    With scope
        .Output "Waveform:points " & Points
        .Output "Waveform:Source " & strChannel
        .Output "Waveform:Preamble?"
        .Enter Preamble()
        .Output "Waveform:Format byte"
        .Output "Waveform:data?"
        ' this gets the IEEE block data
        .Enter ydata, "I1"
    End With

    ' dimension data for the needed size
    ReDim data(Points - 1)
    ReDim time(Points - 1)

    ' scale the data to volts and seconds
    For i = 0 To UBound(ydata)
        data(i) = (ydata(i) - Preamble(9)) * _
                  Preamble(7) + Preamble(8)
        time(i) = ((i - Preamble(6)) * _
                  Preamble(4)) + Preamble(5)
    Next i ' 0  to Ubound of ydata
End Sub
```

Next, you're ready to use the control's embedded automation server to control instruments through VB code. In your code, use a control's commands to produce actions. The two commands you'll most often use are "Output" for sending ASCII strings to instruments and "Enter" for receiving data.

To see a brief description of the "Output" command, the "Enter" command, and any other of the control's commands, press the F2 key or click on "Object Browser" in the "View" menu. Select TMWControl where "All Libraries" is shown. To see details of the IO object, in the menu go to "Project, References" and click on "IO Manager and Utilities". You'll then be able to see the commands that the control exposes with its I/O library. Alternatively, you can place the cursor on the method or property name while viewing VB code and press the F1 key. You will then see a help page for that method or property.

To check that you can communicate with the instrument through code, place a button on the form, add the following code in the button_click subroutine, and run the program. If you receive a response identifying your instrument, then you have communicated with it.

```
Dim reply As String
   scope.Output "*IDN?"
   scope.Enter reply
MsgBox reply
```

This general-purpose control will work for instruments that accept string commands. You can go further by encapsulating code unique to an instrument into your own custom control. You just expose the command GetWavformData, and have it return the data array. You will then have an ActiveX control unique to your instrument. In effect, you'll have created a custom driver for your instrument. (To learn more about how to build your own control, see the article in footnote 3.)

Now that you have code to control your instrument, you can get data and plot a waveform. Scopes can return thousands—or even millions—of waveform data points. To transfer the data quickly, the HP 54600 returns the data as simple bytes in an IEEE 488 block format. The scope also returns a preamble with 10 fields that contain scaling information about the data.

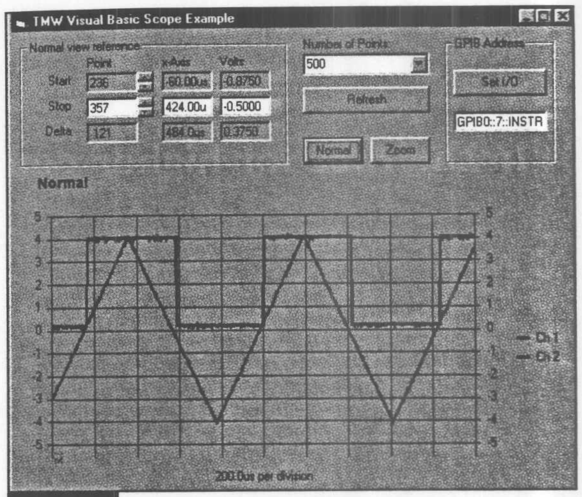
With a scope, you're mostly interested in the y-axis (volts) and x-axis (time) scale factors. With these parameters, you can reconstruct the waveform. **Listing 1** shows you how to use the function GetWaveformData() to retrieve data from the scope.

To get the preamble, use the control's Enter command with an array. Scopes often have a command that let's them send the preamble to the host computer.

```
Dim Preamble(10) As Double
scope.Output "Waveform:
  Preamble?"
scope.Enter Preamble()
```

When you use the control's Enter property with an array that is a numeric data type, the control parses the string returned from the scope into an array. You can check the data in the array with a "For Next" loop and the debug print property. We created a debug button with code that lets us look at the preamble on the form. Look at the data in the "Immediate Window" available from VB's "View" menu.

```
Dim I As Long
For I = 0 to 9
 Debug.print
  Preamble(I)
Next I
```



Start 236 -60.00us -0.8750
Stop 357 424.00u -0.5000
Delta 121 484.0us 0.3750
500
GPIB0::7::INSTR

FIGURE 3 **Add a chart control to your VB form to plot data.**

The waveform's amplitude data (y value) is in IEEE 488 block format, which is a standardized block of data preceded by a header. You can generate the x value because the y data is equally spaced in time, and the preamble tells you the time between each point. The code segments below demonstrate how to get data from the scope. Both examples are acceptable, although the code that uses the "With" statement will run somewhat faster.

```
Dim ydata As Variant
scope.Output "Waveform:
 Format byte"
scope.Output "Waveform:data?"
scope.Enter ydata, "I1"
```

You can also write the code like this:

```
With scope
 .Output "Waveform:Format
   byte"
 .Output "Waveform:data?"
 .Enter ydata, "I1"
End With
```

Graph the Data

Once you can get data from your instruments into arrays, you can use VB's chart control to view the data. Start by referencing the chart control in your VB project; press Ctrl-T and select "Microsoft Chart Control" to place the control in the toolbox. Now, place the chart control on the form. Set the graph as a two-dimensional line graph. The line graph scales the data and draws the needed grids.

You can make these settings in the property pages or in code. We suggest you use code to avoid wrong settings. **Figure 3** shows a VB form containing a chart that plots two signals.

Next, you must set up the data for the chart. The chart control's ChartData(property) command, which loads data into a chart, requires a two-dimensional variant for its input. You can use the first row (row 0) of each column to store labels or legends. If you use this method to feed data to the control, the graph will automatically plot and label the channels by color.

Table 1 shows the structure of an array for plotting data from two scope channels. We prefer to put the time data in the first column but not plot it because the time between samples is

Listing 2. This routine accepts an MSChart and an array of Variants as input to plot the data, set the divisions on the chart, and label the x-axis based on the time data in the first column.

```
Sub MakeGraph(chart As MSChart, Data() As Variant)
' Plots the data to as graph
   Dim points As Long
   Dim ptsPerDivision As Long
   Dim xDivision As Double
   Dim xAxisTitle

   points = UBound(Data)
   ptsPerDivision = points / 10
   xDivision = (Data(2, 0) - Data(1, 0)) * ptsPerDivision
   xAxisTitle = Format$(xDivision) & " sec per division"
   With chart
     .Plot.Axis(VtChAxisIdX).AxisTitle = xAxisTitle
     .Plot.SeriesCollection(1).Position.Excluded = True
     .Plot.Axis(VtChAxisIdX).CategoryScale.Auto = False
     .Plot.Axis(VtChAxisIdX).CategoryScale.DivisionsPerTick
           = ptsPerDivision
     .legend.Location.Visible = True
     .chartType = VtChChartType2dLine
     .ChartData = Data
   End With
End Sub
```

constant. You can tell the graph to hide the first column of data, yet you still have easy access to the data if you need to view it. The chart creates one line for each column of data you want to plot.

The following code creates the array to plot channel one. We added the "+1" in the ReDim statement, which allows for the extra string labels in row 0:

```
Dim data() as Variant
ReDim data(0 To
UBound(ydata) + 1, 1 To 2)
 data(0, 1) = "time"
 data(0, 2) = "Ch 1"
For i = 0 To UBound(ydata)
 data(i + 1, 1) = timedata(i)
 data(i + 1, 2) = ydata(i)
Next i
```

Table 1. Array Structure

"Time"	"Ch 1"	"Ch 2"
x-axis_data(1)	Ch_1_Data(1)	Ch_2_Data(1)
x-axis_data(2)	Ch_1_Data(2)	Ch_2_Data(2)

Once you've created the data array, you can feed the data to the chart with the code in **Listing 2.** The first line of the code below hides Series 1 (column 1, time data). The second line chooses the type of chart, and the last line gives it the data array. The code also scales the grid and annotates the x-axis.

```
MSChart1.Plot.
 SeriesCollection(1).
  Position.Excluded =
  False
MSChart1.chartType
  =VtChChartType2dLine
MSChart1.ChartData = data
```

With the TMWControl and with VB's chart control, you have a powerful set of tools for building quick applications for controlling instruments. While we use these controls mostly for automating tests to evaluate engineering prototypes, you can also use them to automate product tests. *T&MW*

FOOTNOTES

1. Muterspaugh, Helen, "Program Your Scope from a PC," *Test & Measurement World,* October 1998, p. 39, *www.tmworld.com/articles/program_dso_1098.htm*

2. If you have neither OSR2 nor Internet Explorer, you can download Explorer from *www.microsoft.com.* Or, if you prefer not to install Explorer, you can download and install DCOM95 from *www.microsoft.com/com/DCOM/dcom95/dcom1_3.asp.*

3. Freeman, Michael, "Create Your Own ActiveX Controls," *Test & Measurement World,* September 1998, p. 39.

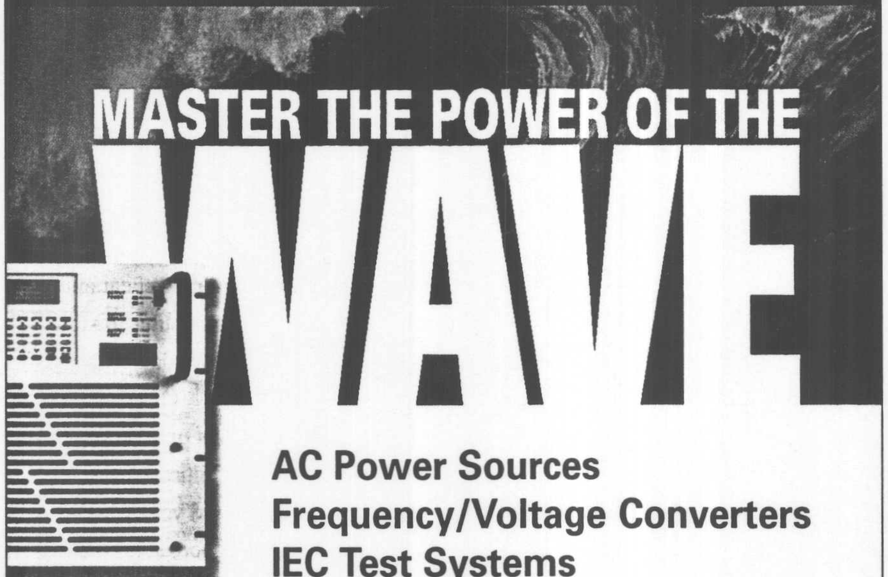
Werner Haussmann *is an R&D project manager with Hewlett-Packard's Electronic Measurements Division in Loveland, CO. He received a B.S.E.E. from Fairleigh Dickinson University.*

Alicia Viskoe *is a software engineer with HP. She has an M.S.E.E. from the University of New Mexico and a B.S.E.E. from the University of Minnesota.*

▶ *If you have technical questions about the TMWControl, contact Werner Haussmann at werner_haussmann@hp.com.*

For quantity reprints of this article, call 847-390-2296.

AC Test Power



MASTER THE POWER OF THE WAVE

AC Power Sources
Frequency/Voltage Converters
IEC Test Systems
Custom Power Products

Pacific Power Source makes it easy to control, meter and analyze AC power with our rugged line of AC Test Power products. Applications include-

- Automatic Test Equipment (ATE)
- Compliance testing (UL, CE, CSA, IEC, Mil Std, TUV)
- Lab and Production testing
- Lighting ballast testing
- Power line disturbance simulation
- 400Hz avionics and shipboard systems

Put us to the test! Call or visit our web site to get a FREE demo copy of Pacific's easy to use software for AC Test Power applications.

www.pacificpower.com

800-854-2433 • 714-898-2691 • 714-898-8076 (FAX)

FREE Demo Software

PACIFIC POWER SOURCE
A **Thermo Voltek** Company

Thermo Voltek · PACIFIC · KeyTek · MILMEGA · COMTEST · GLOBAL